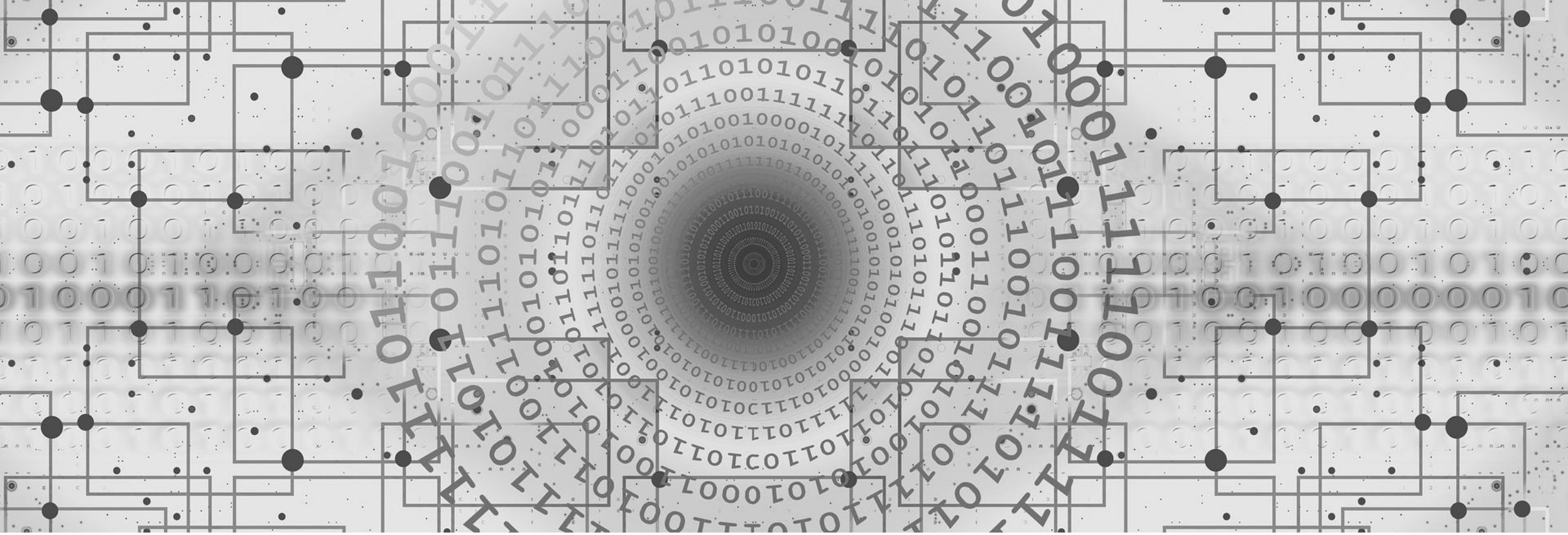


Lecture-14 : FSM State Minimization Techniques

ECE-111

Vishal Karna

Summer 2025

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Motivation For FSM State Minimization

- **Implementing FSM with fewest possible states :**
  - Least number of flip flops required in final hardware generated
  - Reduce the number of gates needed for implementation
    - Final circuit generated potentially becomes less complex
    - In FPGA's, reduced logic gates can reduce ALUT count usage and potentially higher performance of circuit
    - Potential power benefits, as logic gate count is less

- **Two commonly used techniques to reduce FSM states**
  - Implication Chart
  - Row Matching

Implication Chart FSM Minimization Technique

Implication Chart FSM State Minimization

- **Goal :**
  - Identify and combine states that have equivalent behavior

- **Two States are Equivalent :**
  - If they produce the same output for all inputs and their next states are also equivalent

- **Algorithm :**
  1. List All State Pairs
     a. Create a chart of all unique state combinations (e.g., A vs B, A vs C, etc.).
  2. Mark Immediately Non-Equivalent Pairs
     a. If two states produce different outputs for any input, mark them as not equivalent.
  3. Add Implied Pairs
     a. For each unmarked pair, check their next states for each input.
     b. Add those next-state pairs as implied dependencies.
  4. Iteratively Eliminate Conflicts
     a. If any implied pair is already marked non-equivalent, mark the current pair too.
     b. Repeat until no new markings occur.
  5. Merge Equivalent States
     a. Remaining unmarked pairs are equivalent.
     b. Update the FSM by merging these states.

Case Study : Sequence Detector FSM State Minimization

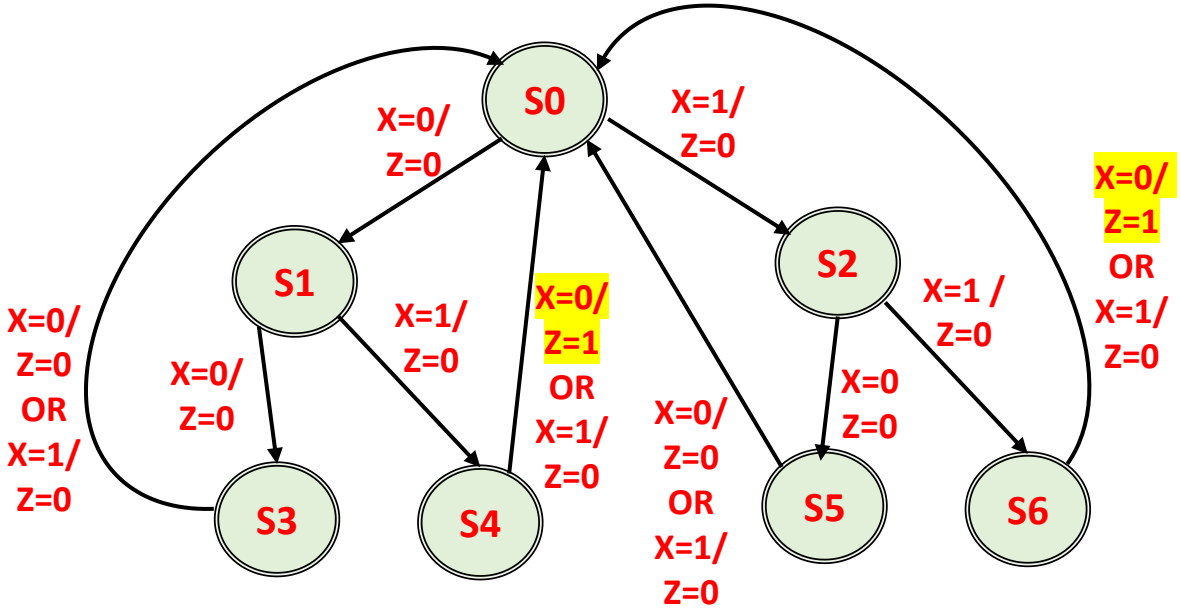
❑ Design Sequence detector FSM which has :

- Single input X, Single output Z
- Output a 1 whenever the serial sequence **010** or **110** has been observed at the inputs

❑ FSM State Transition Table and State Transition Diagram

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

7 states to represent this sequence detector in Mealy FSM without state minimization !

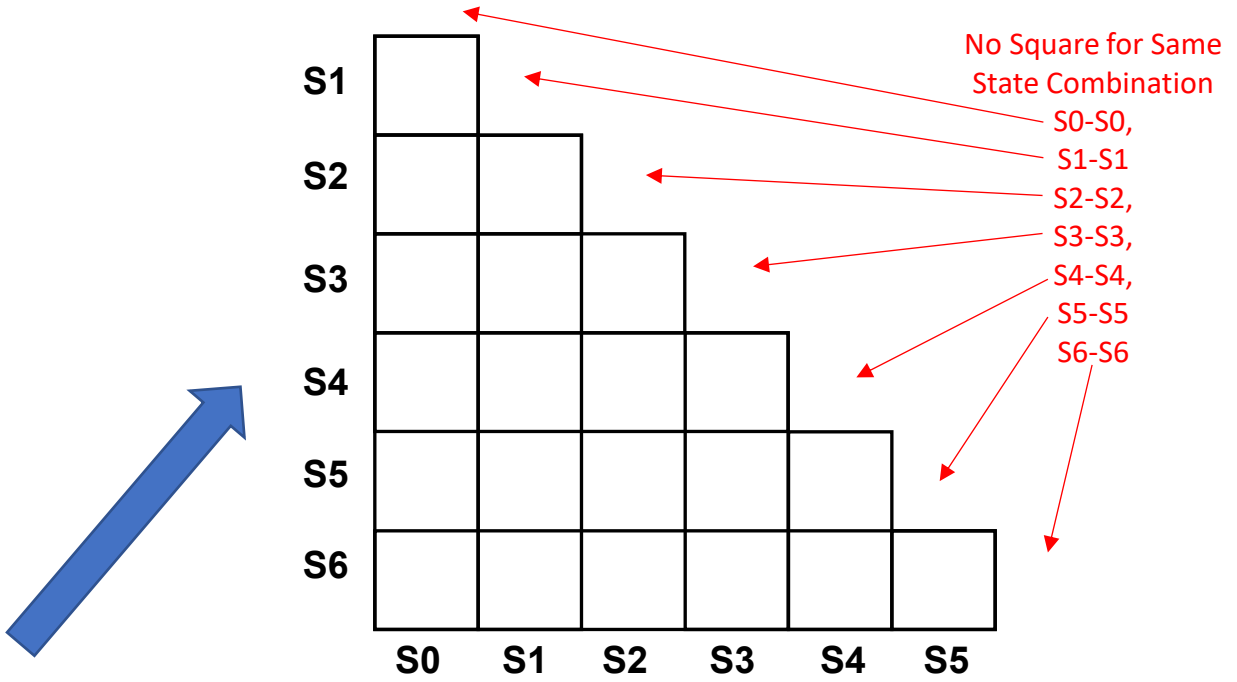


Mealy FSM Diagram for Sequence Detector

Implication Chart Algorithm Steps

- **Step-1 : Construct implication chart, one square for each combination of states taken two at a time**

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

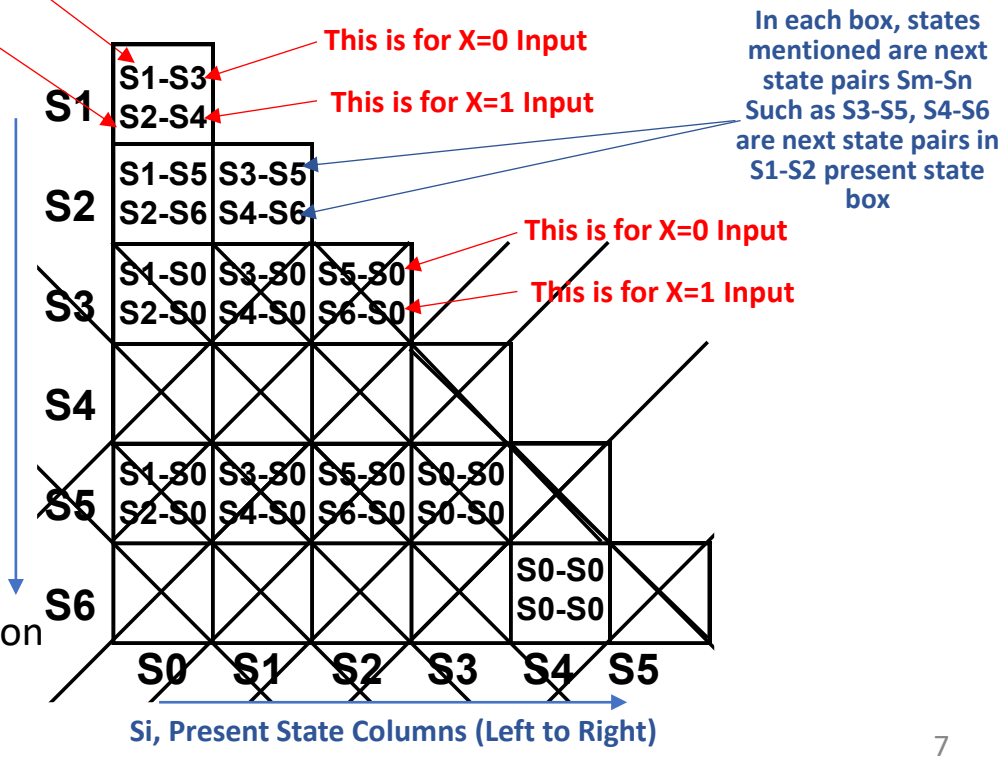


- **There should be square for each of these state combinations :**
  - S0-S1, S0-S2, S0-S3, S0-S4, S0-S5, S0-S6 (**Note :** S0-S0 is same state combination which is not required)
  - S1-S2, S1-S3, S1-S4, S1-S5, S1-S6 (**Note :** S1-S1 not required. S1-S0 is combination is not required as it is covered above)
  - S2-S3, S2-S4, S2-S5, S2-S6 (**Note :** S2-S2 not required. S2-S1, S1-S0 is combination is not required as it is covered above)
  - S3-S4, S3-S5, S3-S6 (**Note :** S3-S3 not required. S3-S2, S3-S1, S3-S0 is combination is not required as it is covered above)
  - S4-S5, S4-S6 (**Note :** S4-S4 not required. S4-S3, S4-S2, S4-S1, S4-S0 is combination is not required as it is covered above)
  - S5-S6 (**Note :** S5-S6 not required. S5-S4, S5-S3, S5-S2, S5-S1, S5-S0 is combination is not required as it is covered above)
  - **Note:** S6-S6 not required. S6-S5, S6-S4, S6-S3, S6-S2, S6-S1, S6-S0 is combination is not required as it is covered above)

Implication Chart Algorithm Steps

- **Step-2 :** Square labeled **Si, Sj** (present states), if outputs differ than square gets cross mark "X". otherwise write down implied next state pairs **Sm-Sn** for all input combinations
  - **Example :**
    - S0 and S4 Rows below have different output 0 and 1 when Input X = 0. So as long as output in two rows do not match for at least one of the input values, then Cross Mark "X".
    - Same applies for S0-S6, S1-S4, S2-S4, S3-S4, S4-S5 present state box combination. Each gets cross mark "X" as for these present state rows, output does not match at least for one of the input values.
    - S0 and S1 Rows have same output for input X = 0, So add S1-S3 next state combination in S0-S1 present state box
    - S0 and S1 Rows have same output for input X = 1, So add S2-S4 next state combination in S0-S1 present state box

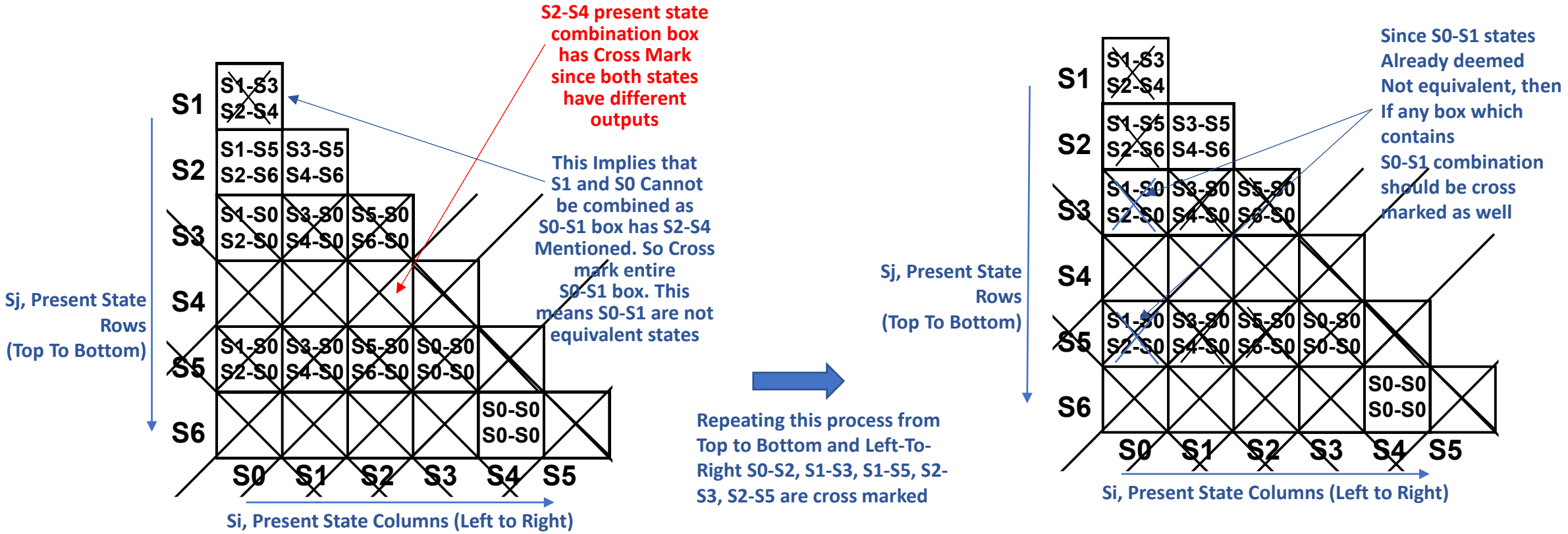
Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0



- Repeat the same process For each state combination box combination
  - For each Column to each Row from Left to Right

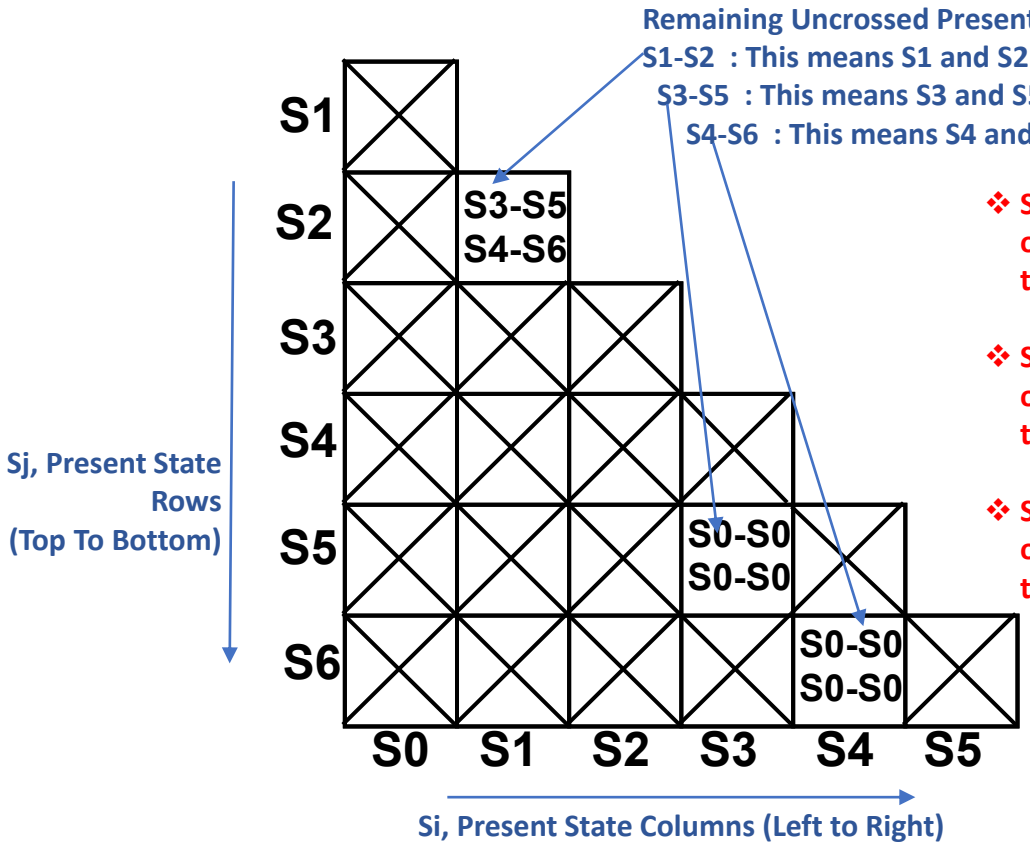
Implication Chart Algorithm Steps

- **Step-3 :** Advance through chart top-to-bottom and left-to-right. If square Si, Sj contains next state pair Sm, Sn and that pair labels square already labeled "X", then Si, Sj is labeled "X".
  - **Example :**
    - S2-S4 (Si-Sj) present state combination box has Cross Mark, and since S2-S4 is mentioned in S0-S1 present state combination box, this implies that S0-S1 present state combination are not equivalent. Hence entire S0-S1 present state combination box should be now Cross Marked.
    - After repeating this process from Top to Bottom and Left-To-Right S0-S2, S1-S3, S1-S5, S2-S3, S2-S5 are cross marked.



Implication Chart Algorithm Steps

- **Step-4 :**For each remaining uncrossed square Si, Sj, these states are deemed as equivalent states. For these equivalent states replace in original state transition table and come up reduced FSM state table
  - S1-S2 are equivalent states as the present state box for S1-S2 is uncrossed
  - S3-S5 are equivalent states as the present state box for S3-S5 is uncrossed
  - S4-S6 are equivalent states as the present state box for S4-S6 is uncrossed



Original State Transition Table

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

- Since S1-S2 are equivalent, remove S2 Row from original table. And Replace everywhere in original table S2 with S1
- Since S3-S5 are equivalent, remove S5 Row from original table. And Replace everywhere in original table S5 with S3
- Since S4-S6 are equivalent, remove S6 Row from original table. And Replace everywhere in original table S4 with S6

Reduced State Transition Table

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_1	0	0
0 or 1	S_1	S_3	S_4	0	0
00 or 10	S_3	S_0	S_0	0	0
01 or 11	S_4	S_0	S_0	1	0

Reduced State Transition Table has 4 states (S0, S1, S3, S4) compared to Original Table which had 7 states (S0 to S6) !

Final Reduced State Transition Table and Diagram

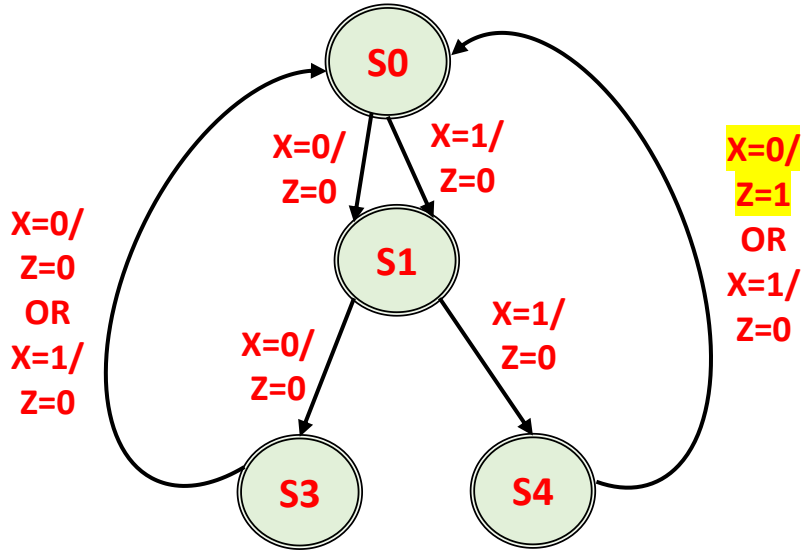
❑ Reduced State Transition Table and State Transition Diagram For Sequence detector

- Transition from S0 to S1 when X=0 followed by S1 to S4 when X=1 followed by S4 to S0 when X=0 will detect **"010"** sequence and output Z will be '1'

- Transition from S0 to S1 when X=1 followed by S1 to S4 when X=1 followed by S4 to S0 when X=0 will detect **"110"** sequence and output Z will be '1'

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_1	0	0
0 or 1	S_1	S_3	S_4	0	0
00 or 10	S_3	S_0	S_0	0	0
01 or 11	S_4	S_0	S_0	1	0

Only 4 states to represent this sequence detector in Mealy FSM using Implication Chart State Minimization Technique !
Original State table had 7 states.



Mealy FSM Diagram for Sequence Detector to detect "010" and "110" Sequence

Row Matching FSM Minimization Technique

Row Matching Technique For FSM State Minimization

- **Goal :**
  - Identify and combine states that have equivalent behavior

- **What are Equivalent States :**
  - For all input combinations, states transition to the same or equivalent states

- **Algorithm :**
  1. Start with original state transition table
  2. Identify states with same output behavior. And, if such states transition to the same next state, they are equivalent
  3. Combine into a single new renamed state
  4. Repeat until no new states can be combined

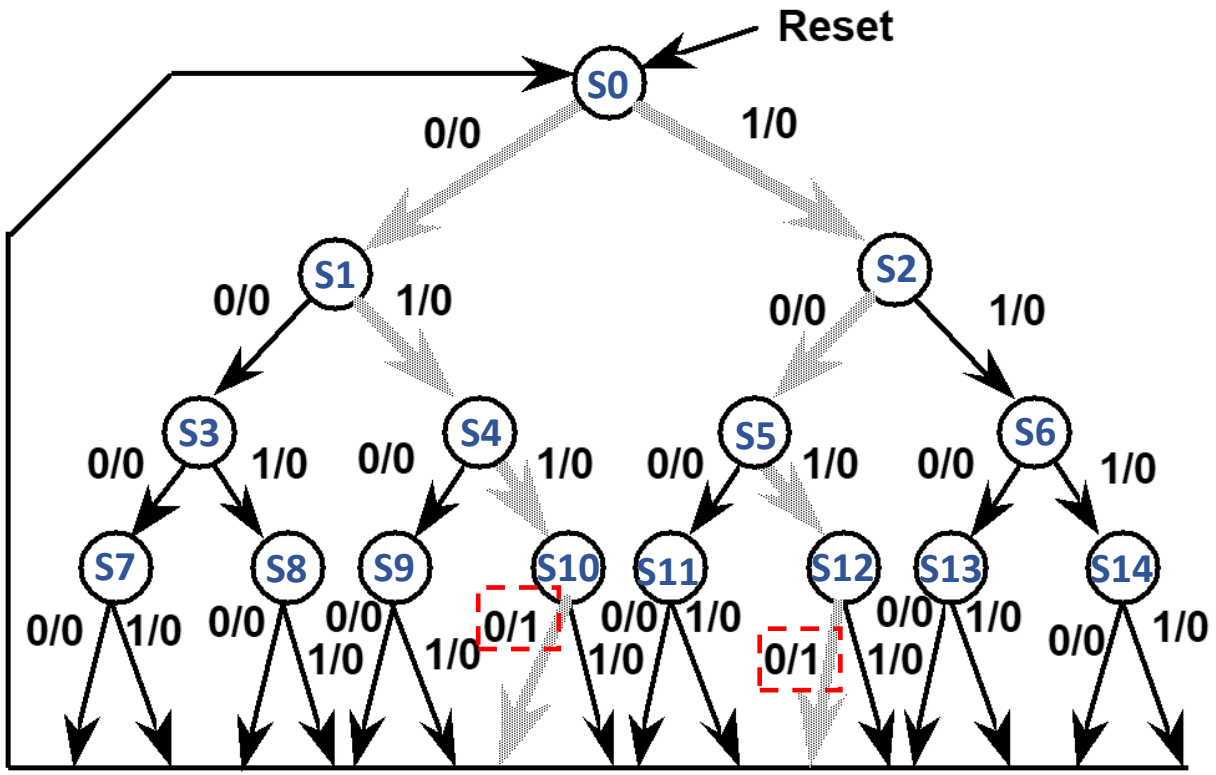
Case Study : Sequence Detector FSM State Minimization

- **Design Sequence detector FSM which has :**
  - Single input X, Single output Z
  - Output a 1 whenever the serial sequence **1010** or **0110** has been observed at the inputs
  - Example :
    - X = 0010 1111 0101 1110 **1010** 0010 1101 **0110** ….
    - Z = 0000 0000 0000 0000 000**1** 0000 0000 000**1**

Row Matching Algorithm Steps

- **Step-1 : Create Initial State Transition Table and State Transition Diagram**

Mealy FSM State Transition Diagram for Sequence Detector



Mealy FSM State Transition Table for Sequence Detector
Total 15 states : S0 to S14

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011	S_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
101	S_{12}	S_0	S_0	1	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

FSM detects 1010 (S0 -> S2 -> S5 -> S12)

FSM detects 0110 (S0 -> S1 -> S4 -> S10)

Row Matching Algorithm Steps

- **Step-2 : Identify present states which have same output behavior and if such states transition to same next state, then these states are equivalent**

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011	S_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
101	S_{12}	S_0	S_0	1	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

Replace S10 and S12 with with new S'10 state name

States S10 and S12 have same outputs and both of these states transition to same next state S0

Hence S10 and S12 are equivalent states and in next step it can be combined into single new renamed state, lets call it as S'10

Also, Relace everywhere S10 and S12 in original FSM stable with new S'10 state name

Row Matching Algorithm Steps

- **Step-3 :** Combine equivalent states into a single new renamed state

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S'_{10}	0	0
10	S_5	S_{11}	S'_{10}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

S10 and S12 are replaced with new S'10 state name

S10 replaced with S'10 and S12 is removed from state transition table as it is merged with S10

Row Matching Algorithm Steps

❑ **Step-4 :** Repeat until no new states can be combined

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S'_{10}	0	0
10	S_5	S_{11}	S'_{10}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

- States S7, S8, S9, S11, S13, S14 have same outputs and all these states, transition to same next state S0
- Hence S7, S8, S9, S11, S13, S14 are equivalent states and in next step it can be combined into single new renamed state, lets call it as S'7
- Also, Replace everywhere S7, S8, S9, S11, S13, S14 in original FSM stable with new S'7 state name

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S'_7	S'_7	0	0
01	S_4	S'_7	S_{10}	0	0
10	S_5	S'_7	S'_{10}	0	0
11	S_6	S'_7	S'_7	0	0
not (011 or 101)	S'_7	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0

S7, S8, S9, S11, S13, S14 in original FSM state table replaced with new S'7 state name

Also, S8, S9, S11, S13,S14 rows are removed as it is equivalent to S7 and combined with S7 and renamed to S'7

Row Matching Algorithm Steps

❑ **Step-4 :** Repeat until no new states can be combined (***Continued ….***)

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S'_7	S'_7	0	0
01	S_4	S'_7	S'_{10}	0	0
10	S_5	S'_7	S'_{10}	0	0
11	S_6	S'_7	S'_7	0	0
not (011 or 101)	S'_7	S_0	S_0	0	0
011 or 101	S'_{10}	S_0	S_0	1	0

- States S3 and S6 have same outputs and both these states, transition to same next state S'7. Hence S3 and S6 are equivalent states and it is combined as S'3 new state
- Similarly, S4 and S5 have same outputs and both these states, transition to same next state S'7 and S'10. Hence S4 and S5 are equivalent states and it is combined as S'4 new state

- Replace everywhere S3, S6, in original FSM stable with new S'3 state name
- Replace everywhere S4, S5, in original FSM stable with new S'4 state name

Input Sequence	Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
Reset	S0	S1	S2	0	0
0	S1	S3'	S4'	0	0
1	S2	S4'	S3'	0	0
00 or 11	S3'	S7'	S7'	0	0
01 or 10	S4'	S7'	S10'	0	0
not (011 or 101)	S7'	S0	S0	0	0
011 or 101	S10'	S0	S0	1	0

- S3, S6, in original FSM stable replaced with new S'3 state name
- S4, S5, in original FSM stable replaced with new S'4 state name
- Also, S6, S5, rows are removed as these are redundant states

Now there are no new states which can be combined. Hence this is now the Final Reduced State Transition Table !

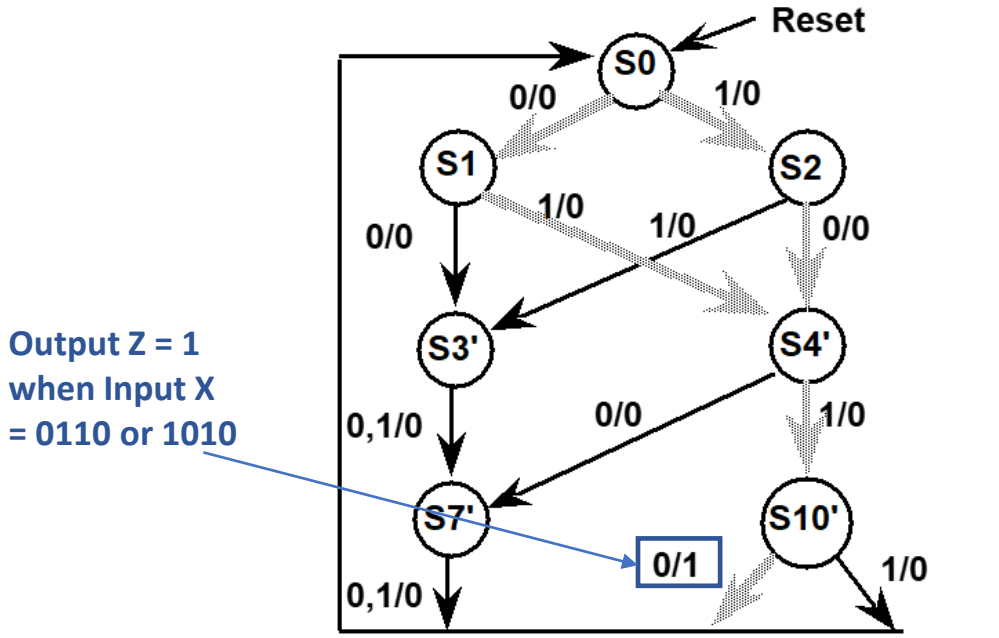
Final Reduced State Transition Table and Diagram

❑ Reduced State Transition Table and State Transition Diagram For Sequence detector

- Transition from S0 -> S2 -> S'4 -> S'10 will detect **"1010"** sequence and output Z will be '1'
- Transition from S0 -> S1 -> S'4 -> S'10 will detect **"0110"** sequence and output Z will be '1'

		Next State		Output	
Input Sequence	Present State	X=0	X=1	X=0	X=1
Reset	S0	S1	S2	0	0
0	S1	S3'	S4'	0	0
1	S2	S4'	S3'	0	0
00 or 11	S3'	S7'	S7'	0	0
01 or 10	S4'	S7'	S10'	0	0
not (011 or 101)	S7'	S0	S0	0	0
011 or 101	S10'	S0	S0	1	0

Only 7 states to represent this sequence detector in Mealy FSM Row Matching Minimization Technique !
Original State Transition Table had 15 states.



Mealy FSM Diagram for Sequence Detector to detect "0110" and "1010" Sequence

References

- For further details on FSM state minimization technique refer to Book : Contemporary Logic Design, 2nd Edition, by Randy H. Katz, University of California, Berkeley & Gaetano Borriello, University of Washington
- Content in these slides are from the above-mentioned book